

Forge a signature to pretend that you are Satoshi.



网络空间安全创新创业实践

Forge a signature to pretend that you are Satoshi

实验报告

王祥宇---202000460053

项目时间：2022 年 7 月 11 日

目录

一、 项目任务	3
二、 实验过程及思路	3
三、 实验测试	4
四、 实验反思与总结	5

一、项目任务

ECDSA – Forge signature when the signed message is not checked

- Key Gen: $P = dG$, n is order
- Sign(m)
 - $k \leftarrow Z_n^*$, $R = kG$
 - $r = R_x \bmod n, r \neq 0$
 - $e = \text{hash}(m)$
 - $s = k^{-1}(e + dr) \bmod n$
 - Signature is (r, s)
- Verify (r, s) of m with P
 - $e = \text{hash}(m)$
 - $w = s^{-1} \bmod n$
 - $(r', s') = e \cdot wG + r \cdot wP$
 - Check if $r' == r$
 - Holds for correct sig since
 - $es^{-1}G + rs^{-1}P = s^{-1}(eG + rP) =$
 - $k(e + dr)^{-1}(e + dr)G = kG = R$
- $\sigma = (r, s)$ is valid signature of m with secret key d
- If only the hash of the signed message is required
- Then anyone can forge signature $\sigma' = (r', s')$ for d
- (Anyone can pretend to be someone else)
- Ecdsa verification is to verify:
 - $s^{-1}(eG + rP) = (x', y') = R'$, $r' = x' \bmod n == r$?
 - To forge, choose $u, v \in \mathbb{F}_n^*$
 - Compute $R' = (x', y') = uG + vP$
 - Choose $r' = x' \bmod n$, to pass verification, we need
 - $s'^{-1}(e'G + r'P) = uG + vP$
 - $s'^{-1}e' = u \bmod n \rightarrow e' = r'uv^{-1} \bmod n$
 - $s'^{-1}r' = v \bmod n \rightarrow s' = r'v^{-1} \bmod n$
 - $\sigma' = (r', s')$ is a valid signature of e' with secret key d

*Project: forge a signature to pretend that you are Satoshi

在任务开始之前仍要完成一些简单的基本运算，比如椭圆曲线上的点的数乘和加法、扩展欧几里得求模逆等等。也要根据上图所显示的过程，实现最基本的ECDSA 签名和验证算法。

二、实验过程及思路

由上图的验证算法我们可以求得：

$$s^{-1}(e*G+rP) \bmod n = (r, s)$$

所以给定 u 和 v ，满足： $u*G+v*P \bmod n=(r, s)$ 。所以此时构造签名 $(r1, s1)$ ，满足下式：

$$r1 = r \bmod n$$

$$s1 = r1 * v^{-1} \bmod n$$

进而构造假的 hash 值，如下：

Forge a signature to pretend that you are Satoshi.

$$e' = r1 * u * v^{-1} \bmod n$$

故这个新的签名对同样的 k 和 d 有效。从而实现了伪造签名。具体实现代码

如下图：

```
#Forge signature when the signed message is not checked
def Pretend(r, s, n, G, P):
    u = random.randrange(1, n - 1)
    v = random.randrange(1, n - 1)
    r1 = Add(Multiply(u, G), Multiply(v, P))[0]
    e1 = (r1 * u * Extended_Euclidean(v, n)) % n
    s1 = (r1 * Extended_Euclidean(v, n)) % n
    Verify_without_m(e1, n, G, r1, s1, P)
```

三、实验测试

为验证上述过程，选择一些参数进行初始化，调用上述函数进行伪造。具体

测试代码如下图：

```
#初始化
a = 2
b = 2
p = 17
m = 'high'
m_1 = "grade"
G = [5, 1]
n = 19
k = 2
d = 5
P = Multiply(d, G) #公钥

#测试签名和验证
print("1. 测试ECDSA签名和验证算法")
r, s = ECDSA_Sign(m, n, G, d, k)
print("签名为:", r, s)
print("验证结果为: ")
ECDSA_Verify(m, n, G, r, s, P)
print('\n')

#Forge signature when the signed message is not checked
print("Forge signature when the signed message is not checked")
Pretend(r, s, n, G, P)
print('\n')
```

Forge a signature to pretend that you are Satoshi.

测试结果图下：

==

1. 测试ECDSA签名和验证算法

签名为: 6 11

验证结果为:

Got it!!!

Forge signature when the signed message is not checked

Got it!!!

四、实验反思与总结

由于先前在 ECDSA pitfalls 项目中已经接触和了解过 ECDSA 的相关攻击和伪造，故此次实验思路比较清晰。通过此次实验，进一步了解了 ECDSA 的伪造攻击，和签名算法可能存在的一些伪造以及安全性隐患，为以后的密码学习奠定了基础。